Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/IL05/000243

International filing date: 02 March 2005 (02.03.2005)

Document type: Certified copy of priority document

Document details: Country/Office: US
Number: 60/548,879
Filing date: 02 March 2004 (02.03.2004)

Date of receipt at the International Bureau: 13 April 2005 (13.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

March 09, 2005

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: ***60/548,879***
FILING DATE: ***March 02, 2004***

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

L. Edelen

L. EDELEN
Certifying Officer

PTO/SB/16 (8-00)
Approved for use through 10/31/2002. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

INVENTOR(S)		
Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Evgeny	KAZAKOV	Naale, ISRAEL
Michael	KAZAKOV	Jerusalem, ISRAEL
Eran	PELEG	Kfar Vradim, ISRAEL

☐ *Additional inventors are being named on the ^ separately numbered sheets attached hereto*

TITLE OF THE INVENTION (280 characters max)

DEVICE, SYSTEM AND METHOD FOR ACCELERATED MODELING

CORRESPONDENCE ADDRESS

Direct all correspondence to:

☒ Customer Number 27130 → *Place Customer Number Bar Code Label here*

Type Customer Number here

OR

☒ Firm or Individual Name	Eitan, Pearl, Latzer & Cohen Zedek, LLP.				
Address	10 Rockefeller Plaza				
Address	Suite 1001				
City	New York	State	New York	ZIP	10020
Country	USA	Telephone	212-632-3480	Fax	212-632-3489

ENCLOSED APPLICATION PARTS *(check all that apply)*

☒ Specification *Number of Pages* 40 ☐ CD(s), Number

☐ Drawing(s) *Number of Sheets*

☐ Application Data Sheet. See 37 CFR 1.76 ☒ Other (specify) postcard

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT *(check one)*

☒ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check or money order is enclosed to cover the filing fees

☒ The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number: 05-0649

☐ Payment by credit card. Form PTO-2038 is attached.

FILING FEE AMOUNT ($): 80.00

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U.S. Government agency and the Government contract number are:

Respectfully submitted,

SIGNATURE

TYPED or PRINTED NAME Mark S. Cohen

TELEPHONE 212-632-3480

Date 02 / Mar / 2004

REGISTRATION NO. *(if appropriate)* 42,425

Docket Number: P-6625-USP

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

Attorney Docket No.: P-6625-USP

UNITED STATES PROVISIONAL PATENT APPLICATION

FOR

DEVICE, SYSTEM AND METHOD FOR ACCELERATED MODELING

BACKGROUND OF THE INVENTION

The Information Technology (IT) world is shifting to model driven approach and related technologies, which allow delivering agile but robust reusable assets, in a manageable form, with reuse of both know-how and tangible artifacts, in a short timeframe and with minimal costs and risks.

This trend got further advance both in adoption by main software vendors and in standardization process led by OMG, W3C and other standardization consortiums. Such standards as Unified Modeling Language (UML), Meta Object Facility (MOF), Common Warehouse Metamodel (CWM), and Model Driven Architecture (MDA) are now adopted and supported by the leading software development suites and IT organizations worldwide.

These industry standards bring a dilemma of aligning of universal and business specific aspects through development process. Standards by nature express the common industry terms which ensure a unified modeling foundation for substantially all participating organizations and thus serve as a basis for unifying methodology as well as an enabler for business-to-business interoperability. On the other hand, a common modeling language – UML – by definition cannot be else but too wide, too common, and too abstract to be usable directly in any domain specific area. In contrast to standards' nature, each business entity may have its own language both in operational activities and in supporting technology infrastructure. Thus there is a need to adopt and customize common modeling languages and modeling tools to be able to answer specific business needs in appropriate terms and on appropriate level of abstraction.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be understood by those of ordinary skill in the art that the invention may be practiced without these specific details. In other instances, well-known methods, procedures, components, units and/or circuits have not been described in detail so as not to obscure the invention.

Metaphor Builder Suite

Metaphor Builder (MB) Suite introduces, for example, creation and usage of UML-based customizable Domain Specific Languages (DSL) on the top of existing modeling standards and leading tools. Such an approach provides flexible meta driven solutions for the whole IT development process from requirements through deployment and operation. MB generates domain specific profiles for the underlying UML tools which then control and guide the development process, providing automatic generation of tangible artifacts from high level technology independent models, as well as automatic creation of meta data databases holding the generated assets in terms of customized language. Metaphor Builder Suite provides enterprises an effective enabler and accelerator of industry adopted standards, existing tools, and cost-saving automation approaches.

It is noted that a part of the discussion herein relates, for exemplary purposes, for UML-based modeling or models. However, the present invention is not limited in this regard, and embodiments of the invention may be used in conjunction with various other suitable types of models, modeling, modeling languages, modeling environments, or modeling tools.

It is noted that the term "domain" is used here in its general sense: domain may refer, for example, to a broad set of problem areas such as banking applications, or manufacturing applications, or to technologies such as the domain of J2EE applications, or it may refer to narrow areas of focus such as the security aspects of an application, or the CBD based development.

In accordance with some embodiments of the invention, language customization may be flexible, agile, oriented for non-programmers and based on standards and modeling tools. It may provide reuse and further specialization of language resources. The Metaphor Builder Suite allows, for example, implementing declarative, zero code, flexible and agile process of UML specialization. The Metaphor Builder Suite may allow various other additional or alternate benefits.

In accordance with some embodiments of the invention, Metaphor Builder Suite (MB) may include, for example, a modeling accelerator based on underlying concepts behind the OMG's Model Driven Architecture (MDA) standard. Embodiments of the invention may comply with the industry's leading standards and may operate as one or more additional layers on top of the sector's existing modeling tools. In some embodiments, MB allows modelers to define their domain specific modeling language and to apply it during the modeling process and/or during the automatic application generation process.

In some embodiments, MB may implement its functionality using an MDA compliant lightweight specialization of the modeling language. In order to express substantially all the required domain specific terms and regulations, in one embodiment, MB may use only, or at least, the accepted standard language notations: Unified Modeling Language (UML) and Object Constraint Language (OCL). In some embodiments, MB provides and manages interoperability between existing modeling tools, and the MDA compatible generating tools that use the results of the modeling process.

In one embodiment, MB may include three high level components, for example: Language Builder, Language Runtime, and Language Metadata Database. These components, in turn, may utilize a set of universal, metadata driven components of Metaphor Framework, for example: Properties Inspector, Configuration Manager, Validation Manager, Process Mentor, Constraints Parser, Generator, Documenter, etc.

MB may treat a Domain Specific Language as a composite entity, able to encapsulate other existing languages, and to be encapsulated itself. The full set of language definitions may

include, for example, domain data types and terms, their properties and relationships, domain specific operations, constraints, behavioral patterns, definitions of recommended modeling process, as well as rules for model validation, transformation, querying, etc.

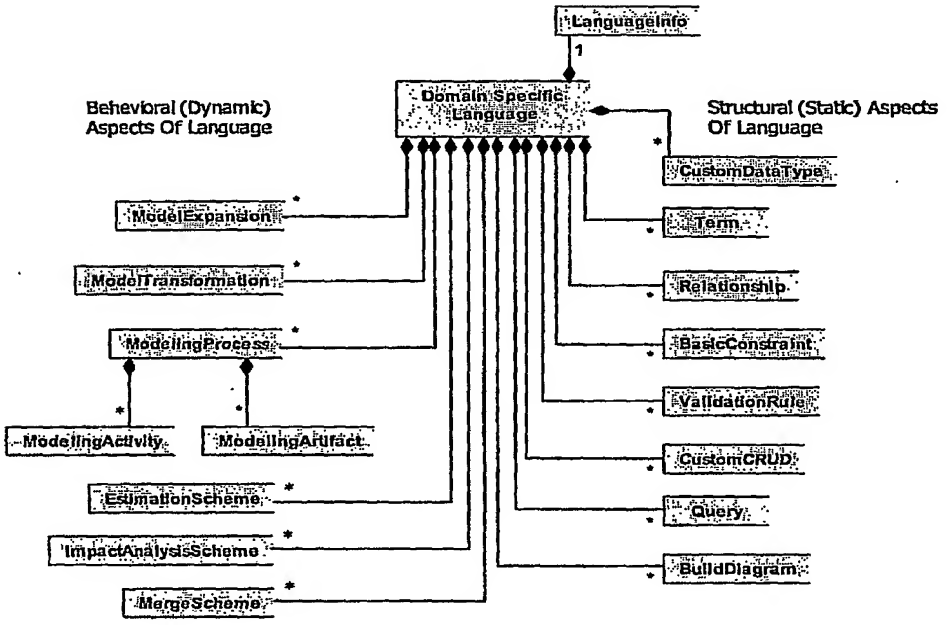

Fig. 1. Structure of Domain Specific Language in accordance with an exemplary embodiment of the invention

Exemplary Component Based Framework

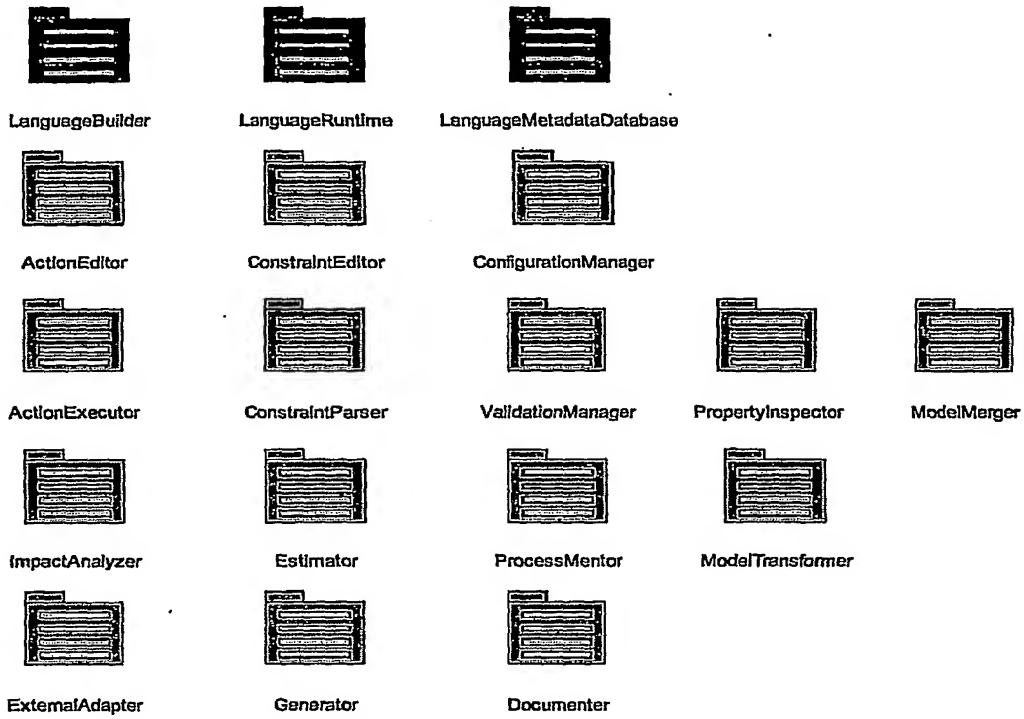

Fig. 2. Components of Metaphor Builder Suite in accordance with an exemplary embodiment of the invention

Component	Description
Action Editor	Action Editor may include a universal, metadata driven component which supports definitions of actions - pieces of additional functionality described in the terms of domain specific language. Actions may be available at modeling time via extended menus and may allow the modeler, for example: - to perform customized CRUDL (Create, Read, Update, Delete, List) operations; - to query the model and to obtain on-line reports; - to create views as pre-defined UML diagrams. Action definition may include action type (what), initial language term (from), resulting language terms (to), as well as filtering criteria and specification of outputs.
Action Executor	Action Executor may include a universal, metadata driven component responsible for providing a custom actions menu and for executing actions, as defined in the language: - customized CRUDL (Create, Read, Update, Delete, List) operations; - querying of entities according to defined criteria; - referencing of relationships and reporting of linked entities; - automatic diagram building.
Configuration Manager	Configuration Manager may be responsible for activation of the Metaphor Builder runtime environment, for example: loading of language definitions and setting of preferences.

Component	Description
Constraint Editor	Constraint Editor may include a universal, metadata driven component which allows definition of constraints for substantially all the entities of the language under construction (language constraints), and for the model elements themselves (model constraints). Language Constraints may populate once-defined regulations for substantially all models which use the language, whereas Model Constraints may be used to refine existing regulations for a particular model element. Constraint Editor allows constraint definitions for substantially all types of language entities - data types, terms, relationships, transformation schemas, process definitions, etc. Constraint Editor may support definitions, for example, in terms of the custom language itself, and may automatically translate them into standard OCL form, which refers to UML's own meta model.
Constraint Parser	Constraint Parser may include a universal, metadata driven component which provides parsing of OCL expressions defining domain specific rules and regulations.
Documenter	Documenter may include a universal model driven generator, which may be intended for automatic creation of documents reflecting the model content and status: reports, proposals, and similar.
Estimator	Estimator is a universal model driven generator performing automatic estimations of costs, resources and risks. It utilizes Estimation Scheme, which is a generally included part of any Domain Specific Language.

Component	Description
External Adapter	External Adapter may include a universal, metadata driven component which may integrate Metaphor Suite with third party MDA compliant generators, automatically providing them with one or more inputs, for example: - definitions of architecturally significant parts of the language (That is, Specification); - the tagged model itself (That is, Mapping). Using these inputs, MDA compliant generators may focus on the generation process, based on the "mapped" UML model. The process may result in different types of tangible artifacts such as, for example, codes, scripts, help files, installed procedures, etc.
Generator	Generator may include a universal, metadata driven component which performs MDA compliant generation process, i.e. automatic building of tangible artifacts from the model entities. A purpose of the built-in Generator is to support the transition from language definition to language supported modeling, for example: - generation of language definitions in XML from the language meta model; - generation of Language related profile installation scripts for a predefined set of generic modeling tools; - generation of initialization and upgrade DDL scripts for the MOF compliant repository of language definitions.
Impact Analyzer	Impact Analyzer may include a universal model driven generator performing automatic impact analysis on the model under construction. It may use Impact Analysis Scheme which may be a part of Domain Specific Language.

Component	Description
Model Merger	Model Merger may include a universal, metadata driven component responsible for: - merging of models; - upgrading of a model if changes in modeling language have occurred. Model Merger may use Merge Scheme which may be a regular part of Domain Specific Language.
Model Transformer	Model Transformer may include a universal, metadata driven component responsible for intra- model and model-to-model transformations: - automatic model expanding using language rules and defaults; - automatic model-to-model transformations based on transformation schemas (part of the language definitions). It may use Transformation Schemas defined, for example, as a part of Domain Specific Language.
Process Mentor	Process Mentor may include a universal, metadata driven component which incarnates modeling flow definitions, as defined in the language, at modeling time and may provide modeler with, for example: - step-by-step wizards, - next activities prompts; - estimation of modeling progress; - phase sensitive helps; - methodology guiding.
Property Inspector	Property Inspector may include a universal, metadata driven component which allows viewing and editing of extended model elements' properties according to definitions made in the language.

Component	Description
Validation Manager	Validation Manager may include a universal, metadata driven component which evaluates constraints definitions as defined in the language, to ensure the model validity. Validation Manager provides, for example, online reporting of inconsistencies, warnings and errors with back referencing to invalid elements.

Exemplary Process

Metaphor Builder Suite may enable and support a MDA compliant process of model-driven development from initial requirements and up to working solution, for example, based on defining and utilizing of domain specific modeling language which expresses domain specific terms, regulations and processes, and provides an abstraction able to orchestrate application development:

(1) A Methodologist defines domain specific modeling language, for example: composes available language recourses (previously defined languages) as well as refines, expands and changes language definitions. The iterative process results in, for example, a substantially full set of language definitions, answering organization or domain needs or objectives.

(2) Methodologist populates language definitions, defined in paragraph (1), to specialize a modeling tool (UML profiles), and to organize a storage for domain specific model entities (metadata database initialization scripts).

(3) An Architect defines and assemblies a framework of universal metadata driven components to be a foundation for any solution able to answer domain specific needs or objectives as specified in paragraph (1).

(4) Architect prepares domain specific generation templates allowing automatic transformation of the models based on domain specific language, defined in paragraph (1), into tangible application artifacts utilizing a framework, defined in paragraph (3)

(5) A Modeler creates models in the terms and by the process defined in domain specific language, built in paragraph (1), using modeling tool specialization, provided by paragraph (2).

(6) Modeler translates domain specific model artifacts, created in paragraph (5) into tangible application artifacts, using domain specific generation templates, built in paragraph (4) and definitions of universal metadata driven components, available from paragraph (3).

(7) Modeler populates domain specific model artifacts, created in paragraph (5), to be available as metadata database initialized in paragraph (2).

(8) Working application, built in paragraph (6) using generation templates defined in paragraph (4) and utilizing universal metadata driven components defined in paragraph (3), may access metadata, created in paragraph (7) and located in metadata database initialized in paragraph (2) which represents model elements, created in paragraph (5) using specialized modeling environment available from paragraph (2), in the terms and by the process of domain specific language defined in paragraph (1). This may allow, for example, to achieve quality of specialized solutions on the top and with cost of generic components.

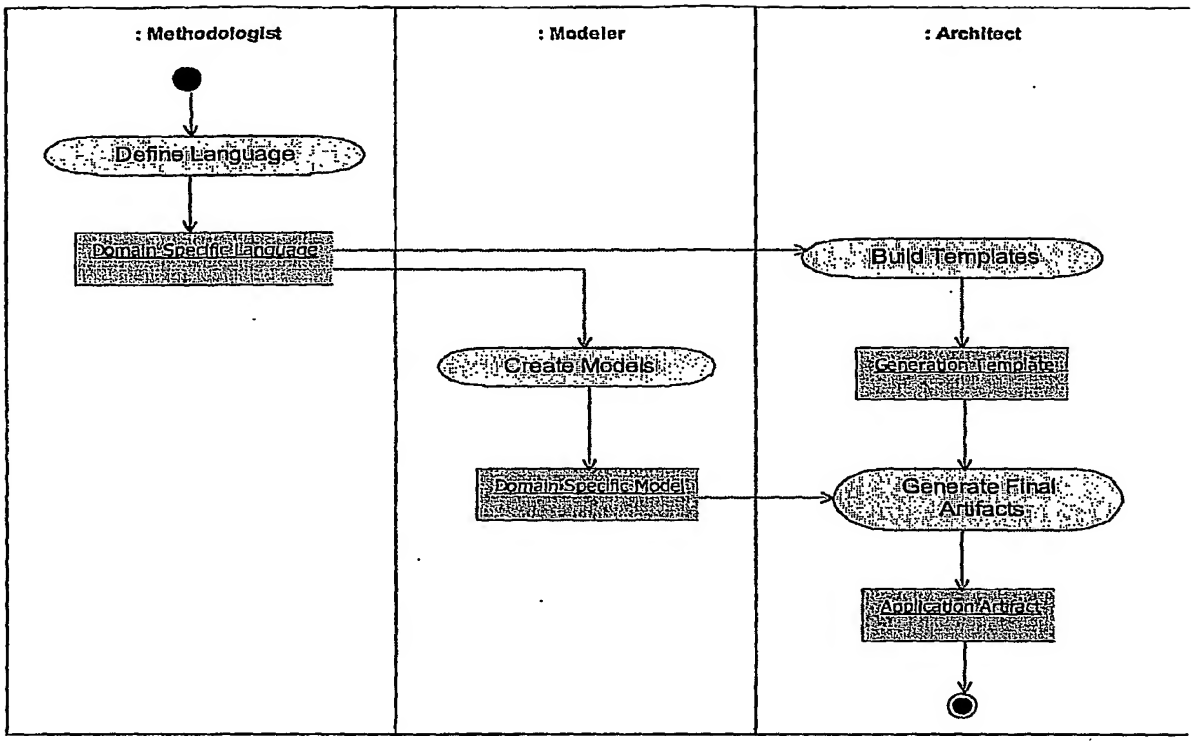

Fig. 3. Metaphor Builder MDA Compliant Process in accordance with an exemplary embodiment of the invention

Exemplary System

In some embodiments, a system may include, for example, one or more modeling stations, and one or more metadata database servers.

In some embodiments, a modeling station may include, for example, a desktop computer, a mobile computer, a laptop computer, a notebook computer, a Personal Digital Assistant (PDA) device, a tablet computer, a server computer, a network, or other suitable computing platform or computing station.

In one embodiment, for example, a modeling station may include at least the following configuration: Intel Pentium IV processor; 30 GigaByte hard disk drive; 256 or 640 MegaByte Random Access Memory; Linux operating system or Microsoft Windows operating system, e.g., Windows 2000 Workstation or higher.

In some embodiments, a metadata database server may include, for example, a desktop computer, a mobile computer, a laptop computer, a notebook computer, a PDA device, a tablet computer, a server computer, a network, or other suitable computing platform or computing station.

In one embodiment, for example, a metadata database server may include at least the following configuration: Intel Pentium IV processor; 80 GigaByte hard disk drive; 512 or 640 MegaByte Random Access Memory; Linux operating system or Microsoft Windows operating system, e.g., Windows 2000 Server or higher.

Other suitable hardware components and/or software components may be used.

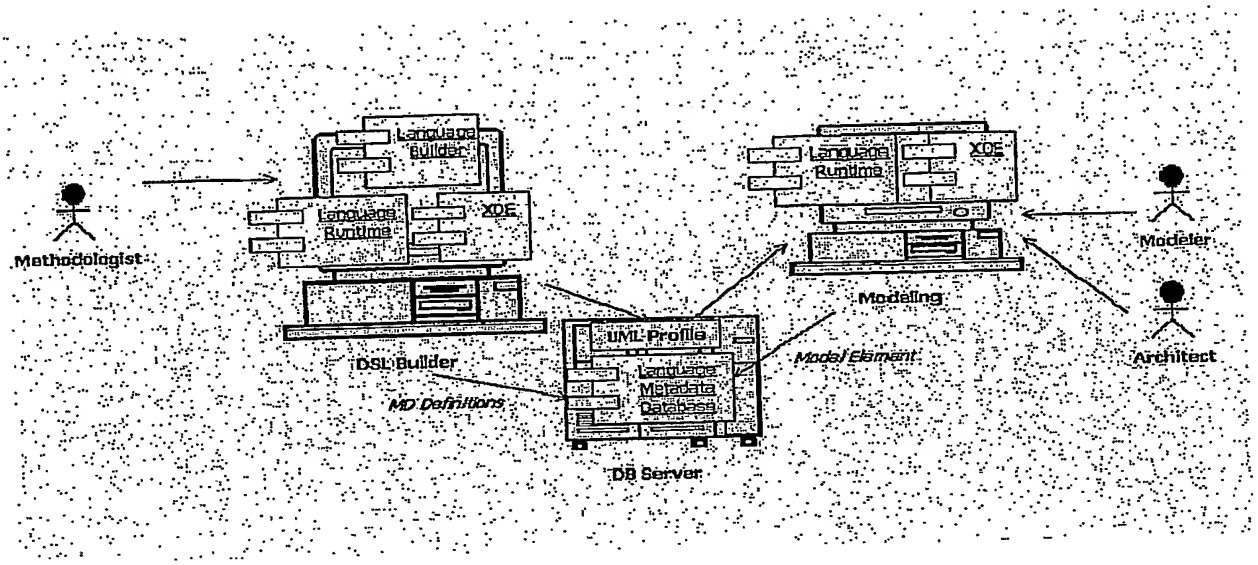

Fig. 4. Metaphor Builder MDA Compliant Process in accordance with an exemplary embodiment of the invention

Language Builder

In accordance with some embodiments of the invention, an exemplary Language Builder may include an MDA compliant meta-modeling tool that allows definition of domain specific modeling languages in the form of light-weight UML specialization. Language Builder defines languages as regular UML models (meta models), using generic modeling tools. To support meta modeling specific functionality, Language Builder may use a specially designed UML Meta Modeling profile, which in conjunction with corresponding pre-built constraints, rules, and data types may constitute a meta modeling language available within Language Builder installation.

Language Builder may treat a language as a composite entity, which is able to encapsulate other existing languages, and to be encapsulated itself. The full set of language definitions may include, for example, domain data types and terms, their properties and relationships, domain

specific operations, constraints, behavioral patterns, definitions of recommended modeling process, as well as rules for model validation, transformation, querying, etc.

Language Builder allows strong properties typing, including pre-built extended sets of data types, values lists (static lookups), and model elements referencing (dynamic lookups). The strong typing may be supplemented by modeling time presentation and editing accessories, as well as by semantic properties grouping.

Language Builder produces, for example, a Domain Specific UML based Language, which encapsulates substantially all the definitions made at the meta modeling stage to be utilized at modeling time. These definitions may be used to support domain specific modeling and to be translated into metadata database definitions.

Define Language

In accordance with some embodiments of the invention, an exemplary Use Case may describe the process of Language Definition, including composition from existing language resources with additional specialization if desired, defining of substantially all types of language elements, and building of output language representations.

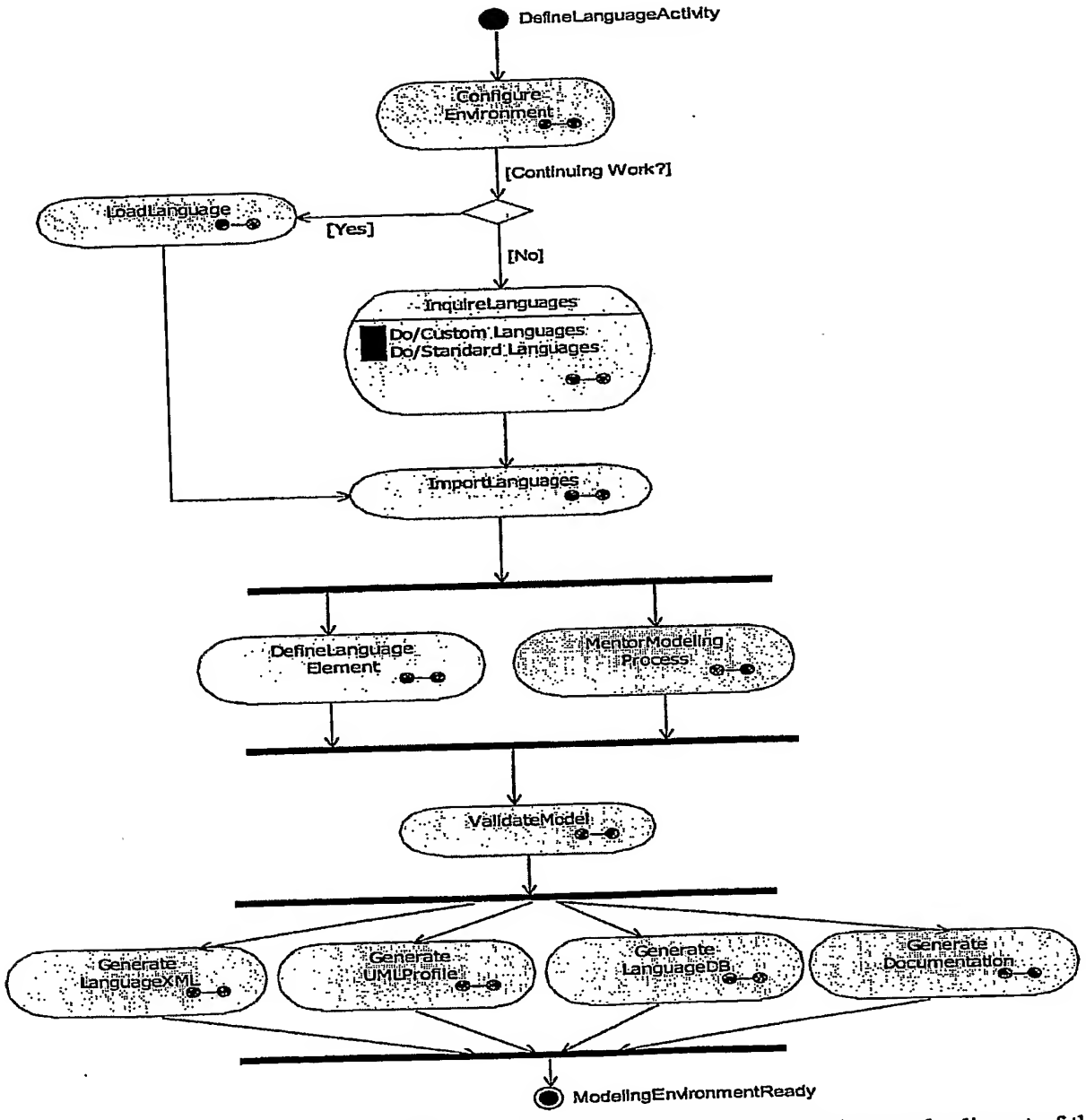

Fig. 5. Define Language Activity Diagram in accordance with an exemplary embodiment of the invention

In some embodiments, substantially each language may include a UML model describing domain terms, their relationships, constraints, permitted operations and recommended flow.

The language definition process may be controlled by a pre-defined meta modeling language, which may have its own terms (for example, "entity as class", "link as association", etc.), constraints (for example, "the term has one and only one UML-mapping stereotype", etc.), validation rules (for example, "no isolated terms", "only precise definitions ", etc.), actions, recommended flow, etc. In one embodiment, this meta modeling language may be, for example, hard coded and incarnated in the meta modeling process by the pre-built UML profile.

A language may optionally include other languages as stereotyped packages. Newly created entities may be interlinked, for example, with those that have come from imported languages.

Activity	Description
Configure Operational Environment	Results in definitions of settings, directories, defaults, etc.
Define Language Elements	Describes repeatable actions of defining of data types and enumerations, language terms, relationships, constraints, rules, actions, recommended process flow, as well as language-level info.

Activity	Description
Discover Existing Languages	Allows inquiry process to select and adopt existing languages to be imported into newly defined one. Metaphor Builder may include, for example, one or more predefined languages which can be integrated into a domain specific language that is under construction to allow additional functionality and behavior - CBD-style organization, testing and similar. Once a language has been included into a language under consideration, it may be open or fully open to customization. In addition, an existing custom language created by MB can be composed into a newly created domain specific language. Substantially each language is stored in a principal external format, as an XML file according to specially designed XML schema. There may be a part of language definition that allows self-explanation to support the inquiry process: language name, version, author, description, domain the language belongs to, list of main terms, etc. The activity results in, for example, one or more existing languages selected to be foundation for the language under construction.
Generate Language XML	Saves language definitions in the principal external format – for example, as an XML file according to specially designed XML schema - Metaphor Builder Schema. Such XML includes substantially all language definitions and serves as the main language exchange unit, for example, to enable language based modeling, to allow composing of this language into newly created one, to load language definitions into MOF repository and so on.

Activity	Description
Generate MOF Repository	Creates either initial scripts or alters existing scripts for database structures of MOF compliant language repository - Metaphor Language Metadata Database. In this way, meta structures of the language may be transformed into repository structures. Populating of the database itself, i.e. language definition entities, can be performed, for example, by loading of language XML files into prepared database tables.
Generate Specification	Generates language related documentation, using (or combining) existing standards, for example: HUNT, RAS, etc.
Generate UML Profile Install	Builds installation scripts of UML profiles for the selected target modeling tool (Rational Rose, Rational XDE, etc) . The language may be translated to one or more UML profiles. Note that profile information does not necessarily include full language definitions, but represents, for example, a subset of language definitions which can be recognized directly by the target modeling tool. The rest of the language definitions, recognizable by Metaphor Language Runtime, may come, for example, from language definition XML file.
Import Other Language Definitions	Imports existing languages, for the first time and/or in upgrade mode, into new language definitions. Substantially every imported language becomes to be package with stereotype "Language" inside the new language meta model.
Load Language	Loads previously saved definitions of the language under construction.

Activity	Description
Mentor Language Building Process	A language optionally contains a process flow definition which describes main modeling activities and corresponding objects of the language terms (modeling artifacts). Such definition is carried out, for example, with a standard activity diagram with or without embedded sub-diagrams and associated objects. To achieve a desired level of precision, special stereotypes for activities, objects and transitions may be used, providing additional fields such as mandatory requirement, comments, links to particular help topics, etc. The definitions described above may allow, for example, customized process mentoring: step-by-step wizards, next step prompts, estimation of modeling progress, working point sensitive help, etc. Reflecting these definitions, a mentor is able to guide the modeling process in substantially all its phases and activities. Language Builder itself may work under language definitions of a predefined Language Definition language. Therefore it may benefit from substantially all the inherent capabilities and types of support, including, for example, meta-modeling time mentoring. Mentoring of Language Building affects substantially all activities involved.
Validate Customer Language	Checks language definitions to ensure, for example, its well formedness: completeness, inter-linkage, etc. The well-formedness rules may be defined as declarative entities inside pre-defined meta modeling language.

Define Language Element

In accordance with some embodiments of the invention, an exemplary Use Case enables definition of language elements, for example: info, data type, term, relationship, constraint, action, etc., as well as refinement of imported language resources and their linking with newly created entities.

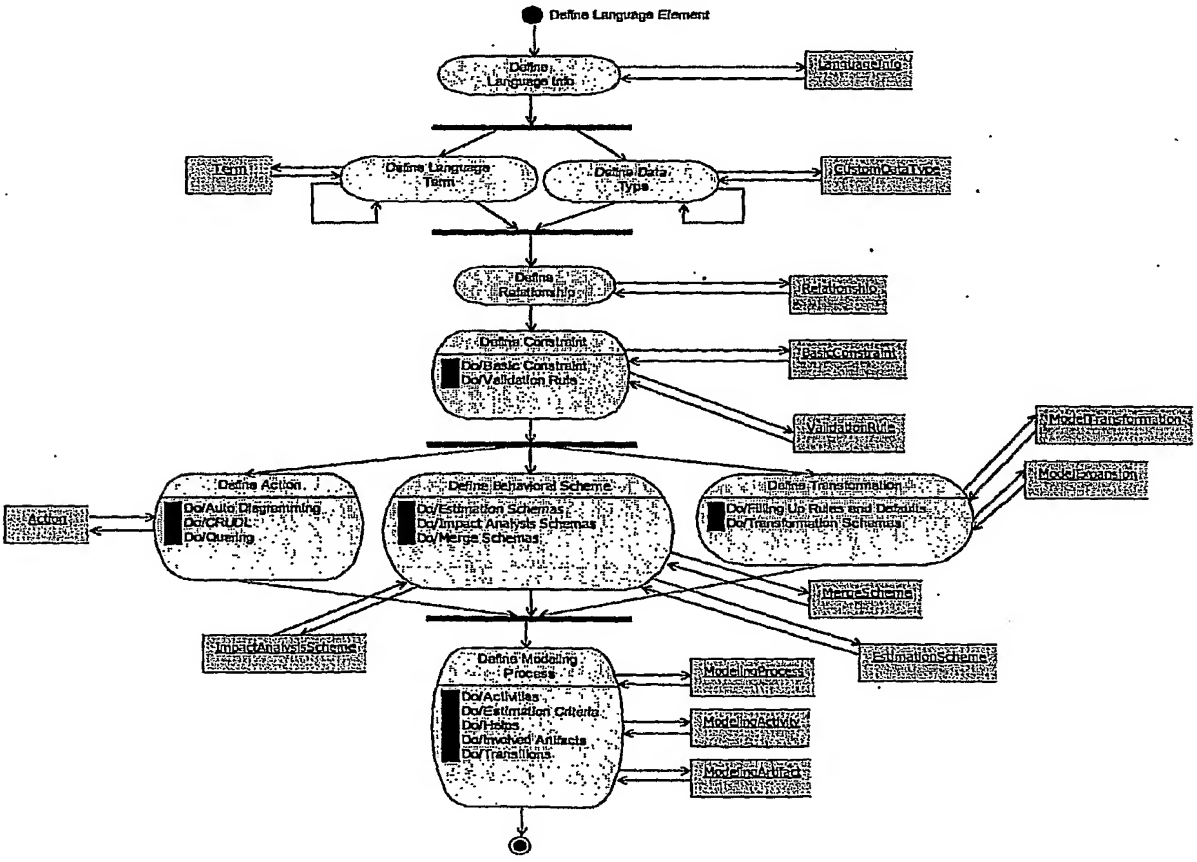

Fig. 6. Define Language Element Activity Diagram in accordance with an exemplary embodiment of the invention

Activity	Description
Define Action	This activity defines, for example, custom Actions to be executed on language terms and relationships: queries, links referencing, automatic diagram building and customized CRUDL (Create, Read, Update, Delete, List) operations. These definitions allow the invocation of custom domain specific functionality at modeling time.
Define Behavioral Scheme	This activity defines, for example, Behavioral Scheme allowing domain specific estimations, merging of models and impact analysis at modeling time.
Define Constraint	This activity defines, for example, constraints for any type of language entity. Constraint definition may be based on standard OCL notation containing three layers: pre-condition, post-condition and invariant. The activity deals, for example, with elementary constraints, related to particular language element, and/or composite constraints, defining validity of the language subset as a whole. These definitions constitute a base for model validation, and a part of the process mentoring, at modeling time.
Define Data Type	This activity defines, for example, language specific data types: primitive or composite data type, enumeration, or reference. These data types are used in definitions of language terms and their properties.
Define Language Info	This activity fills pre-defined language info information, for example: GUID, description, version, author, copyright, etc. Language Info allows languages inquiry and composing.
Define Language Term	This activity defines a term - a main language entity reflecting a domain specific semantic atom. Term definition includes, for example, name, mapping to UML meta class, and set of custom properties. Terms and relationships, as defined in a Language, constitute the main building blocks at modeling time.

Activity	Description
Define Modeling Process	This activity, for example, defines the recommended modeling process deemed to answer organization needs or objectives. These definitions include, for example, Activities, Transitions, Artifacts, Progress Criteria, process sensitive helps and wizards. In conjunction with constraints, these definitions constitute a basis for automatic process mentoring at modeling time.
Define Relationship	This activity defines the relationship between two language terms. Relationship definition includes, for example, name, mapping to UML mechanism (association, attribute-class, element-package), as well as defaults for standard properties and a set of extended properties. Relationships and corresponding terms, as defined in a Language, constitute the main building blocks at modeling time.
Define Transformation	This activity defines a Transformation Scheme allowing either extension of existing model entities by additional properties or creation of new model entities corresponding to existing ones. These definitions enable automatic model transformations at modeling time.

Import Language

In accordance with some embodiments of the invention, an exemplary Use Case allows importing existing language resources into the language under construction. For first time, the resource may be included into newly created Language package. If the resource is represented in the constructed language, it may be re-imported into existing package with analysis of impacts and updating of dependent definitions.

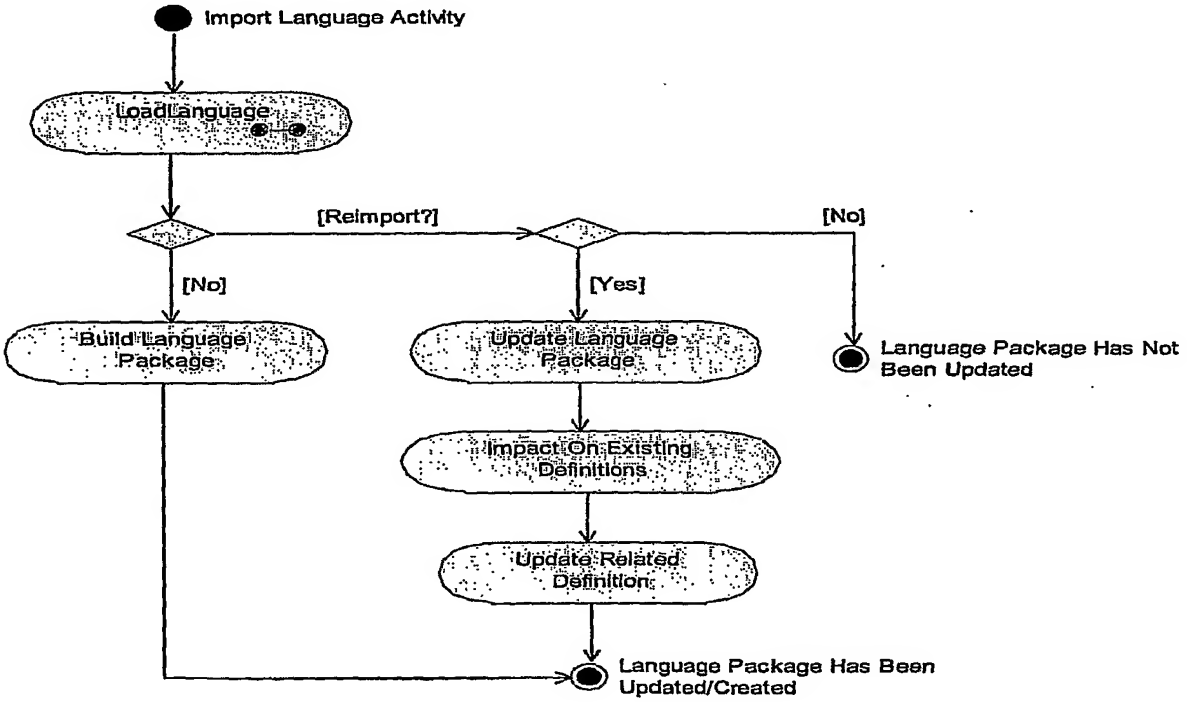

Fig. 7. Import Languages Activity Diagram in accordance with an exemplary embodiment of the invention

Activity	Description
Load Language	This activity loads definitions of the existing language.

Activity	Description
Build Language Package	This activity builds a new language package containing substantially all language definitions.
Impact On Existing Definitions	This activity checks impact of the language package on the rest of the language definitions.
Update Language Package	This activity updates an existing language package with substantially all contained language definitions.
Update Related Definition	This activity updates dependent language definitions.

Inquire Languages

In accordance with some embodiments of the invention, an exemplary Use Case allows, for example, browsing of available language resources, viewing of the resource info, and selection of the resource to be imported into the newly created language.

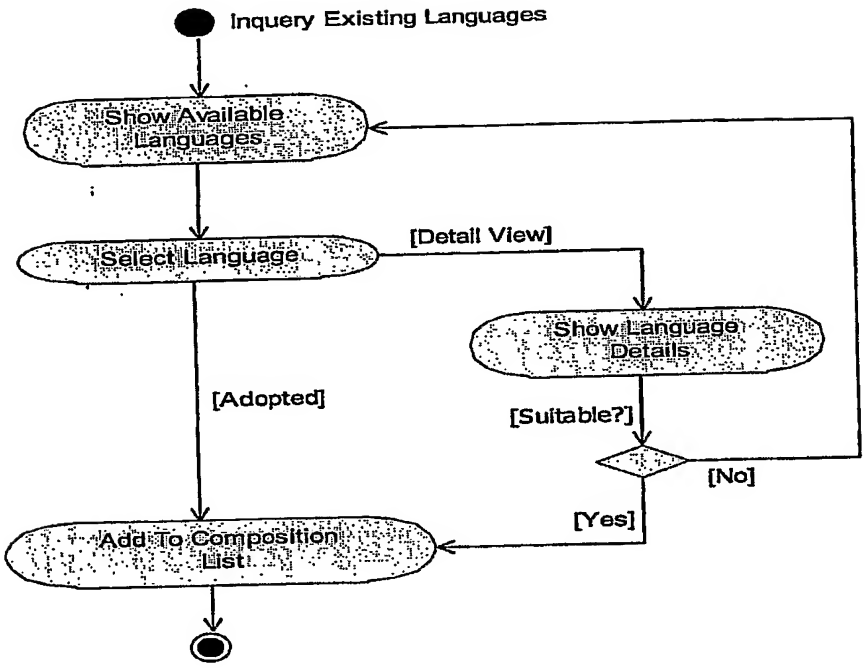

Fig. 8. Inquiry Language Activity Diagram in accordance with an exemplary embodiment of the invention

Activity	Description
Add To Composition List	This activity adds the resources to composition list.
Select Language	This activity selects available language resources.
Show Available Languages	This activity allows browsing of language resources available.
Show Language Details	This activity exposes language info of the selected language resource.

Exemplary Component Based Implementation of Language Builder

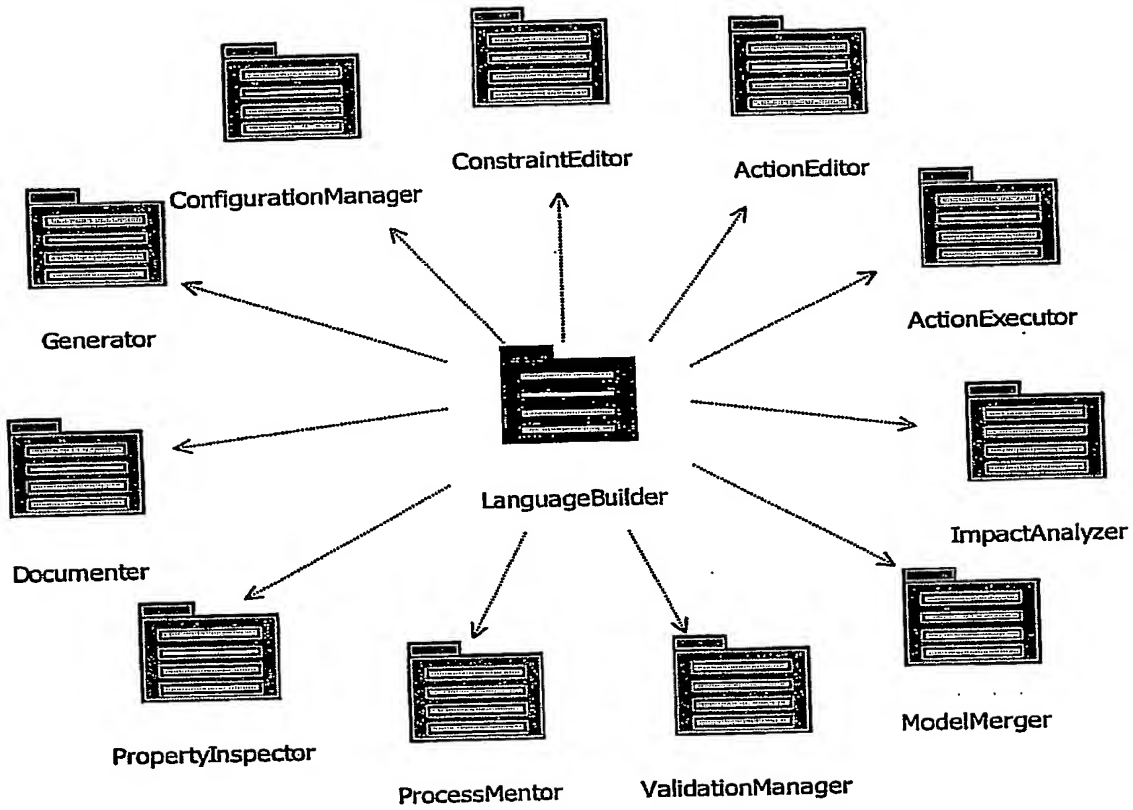

Fig. 9. Language Builder Implementation in accordance with an exemplary embodiment of the invention

Components and their Usage
Constraint Editor allows editing of domain specific language constraints.
Action Editor allows editing of domain specific language actions, as well as supporting on-the-fly definition of actions based on terms of a pre-built Meta Modeling Language.
Generator transforms language definitions into output artifacts: Language Definitions XML, Language UML profiles, and DDL scripts for Metadata Database.

Components and their Usage
Documenter generates domain specific language documentation, based on terms of a pre-built Meta Modeling Language and using pre-built documentation templates.
Validation Manager utilizes constraints information as defined in pre-built Meta Modeling Language to ensure consistency and precision of domain specific language under construction.
Action Executor performs actions as defined in a pre-built Meta Modeling Language or constructed on-the-fly within the selected meta model scope.
Process Mentor ensures recommended meta modeling process is executed as defined in a pre-built Meta Modeling Language.
Configuration Manager enables language definitions and configuration information which may be used to support the language building process.
Property Inspector allows viewing and editing of the extended properties of language elements, as defined in a pre-built Meta Modeling Language.
Impact Analyzer performs analysis of impacts, for example, if one or more parts of language definitions are changed or embedded languages have been re-imported. It utilizes Impact Analysis Scheme included in the pre-built Meta Modeling Language.
Model Merger performs migration of language definitions and/or language-based models if changed versions of embedded sub-languages have been re-imported.

Language Runtime

In accordance with some embodiments of the invention, an exemplary Language Runtime may include a MDA compliant modeling accessory which applies a domain specific language - vocabulary and/or behavior, at modeling time. Language Runtime may be implemented, for example, as an Add-In for a generic modeling tool. It extends the modeling tool functionality by interpretation of additional knowledge, provided by the language definitions, for example:

- viewing and editing of domain specific properties;
- execution of custom actions;
- semantic model validation;
- impact analysis and reporting;
- merges and upgrades;
- model-to-model transformations;
- automatic model-driven estimations of costs, risks, and resources;
- modeling process mentoring;
- exchange with Metadata Database;
- integration with MDA generation tools.

Support Modeling

In accordance with some embodiments of the invention, an exemplary Use case defines a modeling process built on top of organization language definitions. The modeling process includes, for example, establishment of operational environment, maintenance of extended properties, execution of actions, impact analysis, estimations, transformations, merges, validations, version control, as well as model upgrades when underlying language has been changed, and generation of documentation, XMI files and model-driven artifacts.

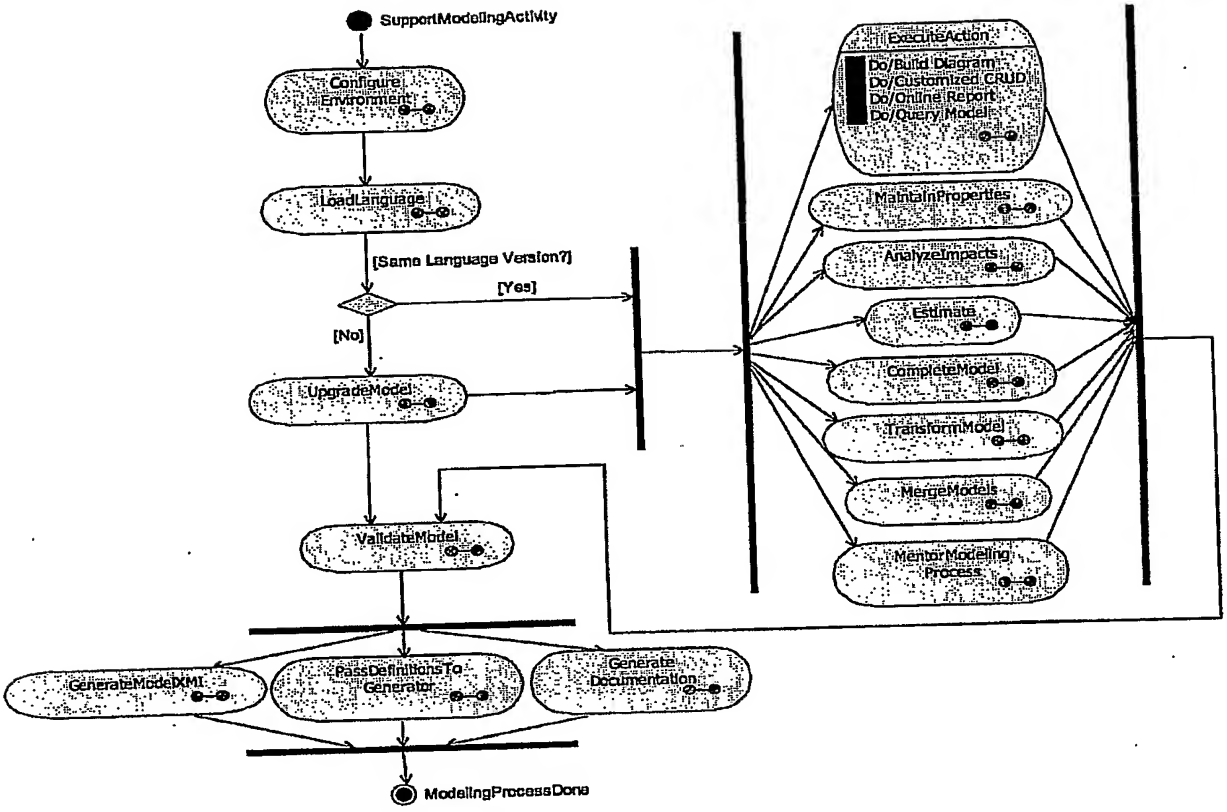

Fig. 10. Support Modeling Activity Diagram in accordance with an exemplary embodiment of the invention

Activity	Description
Complete Model	Expands existing model using expansion templates and defaults defined as part of a domain specific language.
Configure Operational Environment	Results in, for example, definitions of settings, directories, defaults, etc.
Execute Action	Incarnates substantially every action, defined in the domain specific language definition phase, at modeling time and allows user, for example: - to perform customized CRUDL (Create, Read, Update, Delete, List) operations; - to query model and to obtain on line reports; - to create views as pre-defined UML diagrams.
Generate Documentation	Generates model documentation, using (or combining) existing standards, for example: HUNT, RAS, etc.
Generate XMI	Saves the model as standard XMI file extended by language related properties.
Load Language Definitions	Loads previously made language definitions from the persistent storage. Primary storage may include, for example, a language XML file, while in advanced cases it may include MOF compliant repository for language definitions.
Maintain Properties	Allows viewing and editing of extended model properties, as defined in domain specific language, including, for example, strong properties typing, logical grouping, look ups, etc.
Mentor Modeling Process	Performs customized modeling process mentoring, for example: step-by-step wizards, next step prompts, estimation of modeling progress, working point sensitive helps, etc. Mentoring may affect, for example, substantially all activities involved in the modeling process.

Activity	Description
Merge Models	Performs models merging, for example, based on one or more schemas defined in domain specific language.
Pass Definitions To Generator	Generates tangible artifacts from the model, for example: scripts, codes, helps, etc.
Perform Costs and Resources Estimation	Allows invocation of estimation procedures, defined in a domain specific language, and to obtain corresponding online reports.
Perform Impact Analysis	Allows invocation of impact analysis procedures, defined in a domain specific language, and to obtain corresponding online reports.
Transform Model	Performs model-to-model transformations, for example, as defined in a domain specific language.
Upgrade Model	Upgrades an existing model reflecting changes made in language definitions.
Validate Model	Checks the model to ensure, for example, its well formedness and correspondence to language definitions. The validation rules may be defined as constraints inside the customer modeling language.

Exemplary Component Based Implementation of Language Runtime

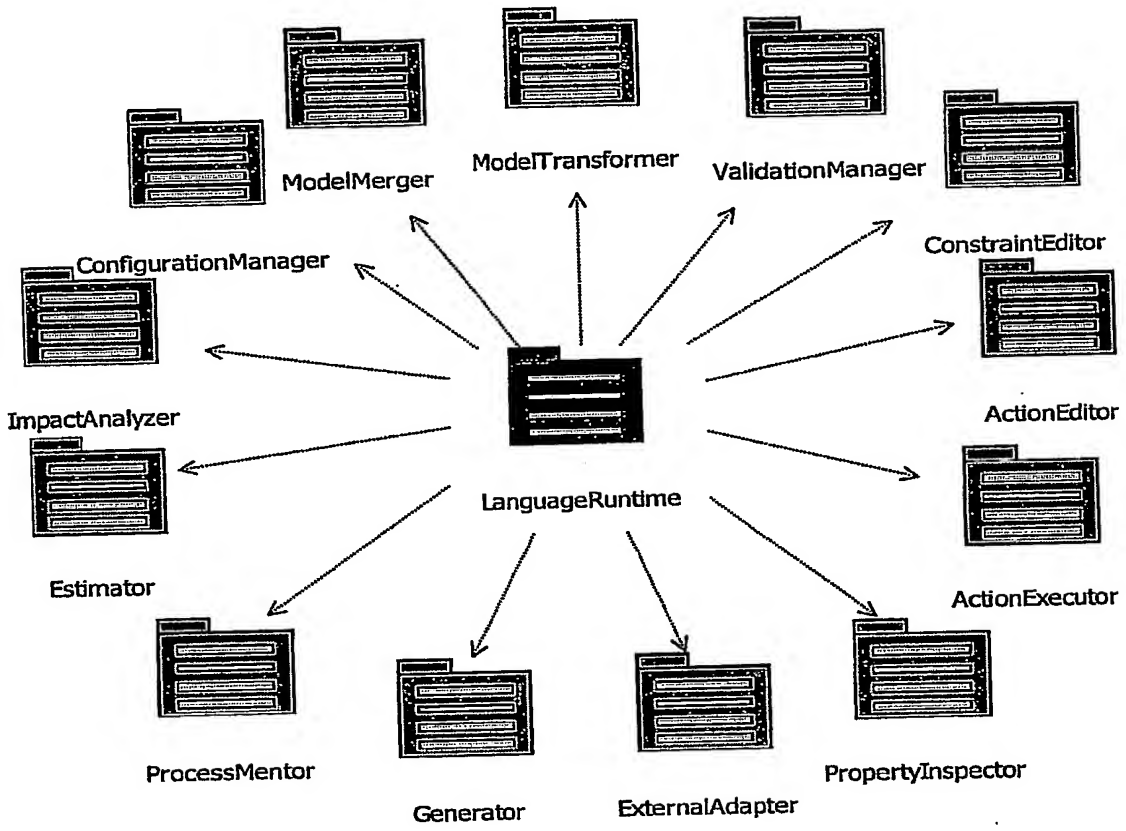

Fig. 11. Language Runtime Implementation in accordance with an exemplary embodiment of the invention

Components and their Usage
Action Editor allows on-the-fly definition of custom actions based on language terms.
Generator allows export of domain specific models to XMI format.
External Adapter enables integration with 3rd party MDA generators.
Validation Manager utilizes constraints information defined in a language to ensure, for example, model consistency and precision. It may validate the whole model or a selected part of it, and may report substantially all inconsistencies found.
Action Executor performs actions, for example, as they are defined in a language or constructed on-the-fly within the selected model scope.
Model Transformer performs model expansions and transformations according to one or more transformation schemas defined in a language.
Model Merger supports merges of domain specific models as well as upgrades, for example, when the underlying language was changed.
Configuration Manager provides language definitions and configuration information which may be used to support domain specific modeling.
Impact Analyzer allows automatic impact analysis and reporting based on a special sub-language designed for impact analysis purposes.
Estimator allows model-driven estimations of costs, resources, and risks.
Process Mentor ensures recommended modeling process as defined in a language, for example: checks model status, detects and warns about conflicts, proposes activities to be performed, estimates percentage of tasks execution.
Property Inspector allows viewing and editing of the extended properties, as defined in domain specific language, including, for example, logical grouping of properties, strong type support, different kinds of lookups, appropriate presentation and editing controls, and just-in-time validations.
Constraints Editor allows on-fly definition of constraints to particular model elements.

Language Metadata Database

In accordance with some embodiments of the invention, an exemplary Language Metadata Database may include a MDA compliant metadata repository which may provide persistence to domain specific meta models and/or models themselves. Language Metadata Database maps meta models to relational tables and can be implemented, for example, using a SQL Database Server or other suitable database formats. Language Metadata Database provides, for example, MOF compliant interfaces to stored entities. In order to support a domain specific language or model without a previous knowledge about its structure, the Language Metadata Database may implement the MOF reflective interfaces - a set of generic interfaces based on common UML terms and allowing step-by-step self-explanation of meta structures at runtime. In some embodiments, Language Metadata Database may support data maintenance functionality for stored metadata entries, as well as XMI import/export.

Access Meta Definitions

In accordance with some embodiments of the invention, an exemplary Use Case allows the accessing of meta data stored in repository, for example: establishing of connection, inquiring of meta data, discovering of meta data structure, selection of meta data, performing CRUDL (Create, Read, Update, Delete, List) operations on meta data entities as well as meta data transformations and XMI exchange.

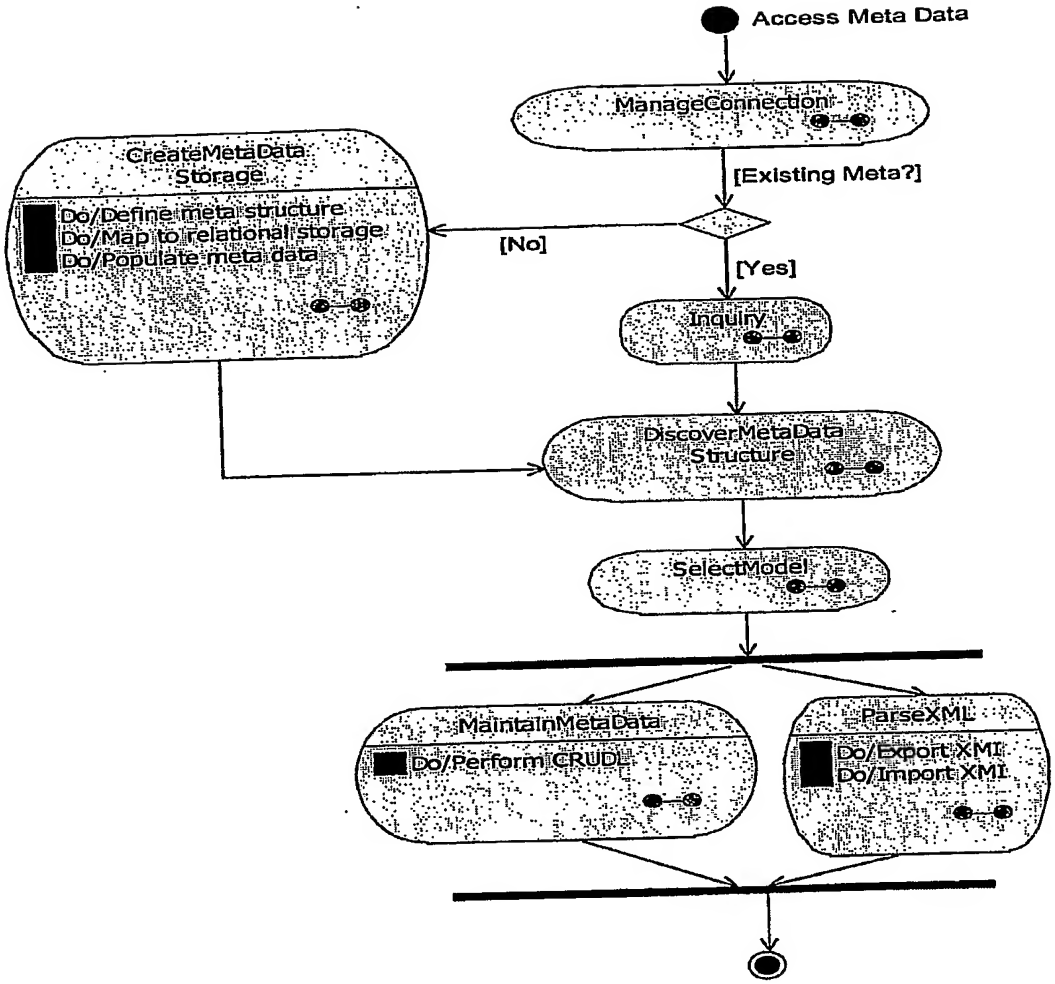

Fig. 12. Access Meta Definitions Activity Diagram in accordance with an exemplary embodiment of the invention

Activity	Description
Query Meta Model	Allows to inquiry meta models stored in repository.
Create Meta Model Storage	Builds meta data storage including, for example, meta definitions, mapping to relational structures and population of meta data.
Discover Meta Model Structure	Discovers meta model structure, for example, using MOF reflective interfaces.
Establish Connection	Establishes connection with the repository.
Exchange Meta Model	Performs XMI import/export of meta data.
Maintain Meta Data	Allows CRUDL (Create, Read, Update, Delete, List) operations on meta model entities.
Select Meta Model	Allows selection of meta model section.

Exemplary Component Based Implementation

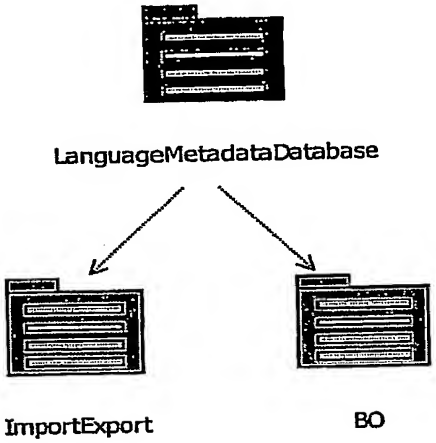

Fig. 13. Language Metadata Database Implementation in accordance with an exemplary embodiment of the invention

Components and their Usage
Import/Export supports XMI exchange.
BO provides basic persistence functionality.

Some embodiments of the invention may be implemented by software, by hardware, or by any combination of software and/or hardware as may be suitable for specific applications or in accordance with specific design requirements.

Embodiments of the invention may include units and/or sub-units, which may be separate of each other or combined together, in whole or in part, and may be implemented using specific, multi-purpose or general processors or controllers, or devices as are known in the art.

Some embodiments of the invention may include buffers, registers, stacks, storage units and/or memory units, for temporary or long-term storage of data or in order to facilitate the operation of a specific embodiment.

Some embodiments of the invention may be implemented, for example, using a machine-readable medium or article which may store an instruction or a set of instructions that, if executed by a machine, for example, by computing station or a processor, or by other suitable machines, cause the machine to perform a method and/or operations in accordance with embodiments of the invention. Such machine may include, for example, any suitable processing platform, computing platform, computing device, processing device, computing system, processing system, computer, processor, or the like, and may be implemented using any suitable combination of hardware and/or software. The machine-readable medium or article may include, for example, any suitable type of memory unit, memory device, memory article, memory medium, storage device, storage article, storage medium and/or storage unit, for example, memory, removable or non-removable media, erasable or non-erasable media, writeable or re-writeable media, digital or analog media, hard disk, floppy disk, Compact Disk Read Only Memory (CD-ROM), Compact Disk Recordable (CD-R), Compact Disk Re-Writeable (CD-RW), optical disk, magnetic media, various types of Digital Versatile Disks (DVDs), a tape, a cassette, or the like. The instructions may include any suitable type of code, for example, source code, compiled code, interpreted code, executable code, static code, dynamic code, or the like, and may be implemented using any suitable high-level, low-level, object-oriented, visual, compiled and/or interpreted programming language, e.g., C, C++, Java, BASIC, Pascal, Fortran, Cobol, assembly language, machine code, or the like.

While certain features of the invention have been illustrated and described herein, many modifications, substitutions, changes, and equivalents may occur to those skilled in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes as fall within the true spirit of the invention.

CLAIMS

What is claimed is:

1. A device substantially as described in any part of the specification and/or as illustrated in any of the drawings.

2. A system substantially as described in any part of the specification and/or as illustrated in any of the drawings

3. A method substantially as described in any part of the specification and/or as illustrated in any of the drawings.